REMARKS

Claims 1, 4, 40, 43, 71 and 73 have been amended, and claims 3, 42 and 72 have been canceled. Claims 1, 2, 4-41, 43-71, and 73-90 are pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Provisional Double Patenting Rejection:

The Examiner provisionally rejected claims 1-90 under the judiciary created doctrine of obviousness-type double patenting as being unpatentable over claims 1-66 of co-pending Application No. 09/663,564. If this rejection becomes non-provisional, Applicants will either present arguments traversing the rejection or file a terminal disclaimer.

Section 102(e) Rejection:

The Examiner rejected claims 1-4 and 6-90 under 35 U.S.C. § 102(e) as being anticipated by Johnson ("XML JavaBeans Integration"). Applicants respectfully traverse this rejection for at least the reasons presented below. Applicants respectfully traverse this rejection for at least the reasons presented below.

Regarding claim 1, Johnson fails to disclose **processing the first object into an intermediary hash table representation of the first object, wherein at least one entry of the intermediary hash table representation includes: a hash key including a name of an instance variable of the first object; and a value for the instance variable**. Instead, Johnson teaches converting JavaBean objects to XML using an intermediary Document Object Model (DOM) tree structure (Johnson, page 3, paragraphs 1 – 8). Johnson's DOM tree structure does not include, nor utilize, a hash table representation in which at least one entry has a hash key including a name of an instance variable of the object. Instead, as is well understood in the art, one navigates a DOM through its tree structures, such as by following links (e.g., parent, child and sibling links) between the

nodes of the DOM tree. Johnson does not mention or describe using an intermediary *hash table* representation of an object or about a hash key including a name of an instance variable of the first object.

Furthermore, the Examiner mistakenly equates Johnson's Document Template Definition (DTD) as an intermediary representation. However, even under the Examiner's interpretation of Johnson that equates Johnson's DTD as an intermediary representation, Johnson still fails to disclose an intermediary *hash* table representation or that at least one entry of the intermediary hash table includes a *hash key including a name of an instance variable* of the first object. Instead, Johnson's "DTD tells the parser how to check the JavaBean's XML for correctness". Johnson also states, "If you use a validating parser with this DTD, you can be certain that any XML tree you read from the file follows the rules for a well-formed XMLBean" (Johnson, page 5). Thus, Johnson specifically teaches that his DTD defines the structure of a properly formatted XMLJavaBean document. Johnson's DTD does not include instance names or values for variables of an object. Johnson's DTD is simply not an intermediary representation, contrary to the Examiner contention.

Thus, Johnson clearly fails to disclose all the limitations of claim 1. As such, Johnson cannot be said to anticipate (e.g., under 35 USC § 102) claim 1. The Examiner's rejection is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 40 and 71.

Regarding claim 12, Johnson fails to disclose a virtual machine receiving a data representation language representation of a first computer programming language object **from a first process**, a decompilation process of the virtual machine generating the object from the data representation language representation of the object, and the decompilation process of the virtual machine providing the first object **to a second process** executing with the virtual machine. Even a cursory read of Johnson clearly demonstrates that Johnson's XMLJavaBeans class is intended to provide a single process the ability to transform a JavaBean in memory into an XML document and vice versa.

Nowhere does Johnson mention anything about receiving an XML representation of an object *from a first process* and returning an object generated from the XML representation *to a second process*, which Johnson would have to teach in order to disclose all the limitations of Applicants' claim 12. In fact, the interface to Johnson's XMLJavaBeans class is clearly configured to return the results (e.g., the object generated from an XML document) to the same method, and hence to the same process, that called the XMLJavaBeans interface (See, e.g., XMLBeanReader method definition, page 9).

Without some specific teaching by Johnson regarding receiving a data representation language representation of a first computer programming language object *from a first process* and providing an object generated from the data representation language representation *to a second process* executing in within the virtual machine, Johnson cannot be said to anticipate (e.g., a rejection under 35 U.S.C. § 102) claim 12.

Thus, the rejection of claim 12 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claim 78.

Regarding claim 25, Johnson fails to disclose **generating a message in the data representation language that includes the data representation language representation of the object and sending the message to a second process**. The Examiner rejected claim 25 relying only the fact that Johnson teaches converting JavaBeans to XML generally without citing any portion of Johnson that discloses, or even discusses, generating a message in a data representation language that includes a data representation language representation of a computer programming object. Instead, Johnson teaches generating an XML document file from a JavaBean object. Nowhere does Johnson mention anything about generating and sending XML representations of objects in XML messages, as would be required for Johnson to anticipate claim 25. In contrast, Johnson teaches an XMLJavaBeans interface that is configured to read and write to files (e.g., on disk drives) not to be send included in data representation language messages.

Johnson fails to disclose all of the specific limitations recited in, and therefore cannot be said to anticipate, Applicants' claim 25. Thus, the rejection of claim 25 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 62 and 84.

Regarding claim 34, Johnson fail to disclose **a first process receiving a message in a data representation language from a second process, wherein the message includes information representing a computer programming language object**. As with the rejection of claim 25, the Examiner rejected claim 34 relying only on the general fact that Johnson teaches the ability to "transform a JavaBean in memory into an XML document" and to "read and write JavaBeans objects as XML documents" and to "transform an XML document … into a running JavaBean", citing the first paragraph of Johnson. However, nowhere does Johnson mention anything about receiving a message in a data representation language that including information representing a computer programming language object. Instead, Johnson describes an XMLJavaBeans interface that is clearly (and unequivocally) configured to read XML documents as files (e.g., from a disk drive). Johnson does not mention receiving XML, or any other data representation language, messages including representations of objects.

The Examiner has not provided any explanation that demonstrates how Johnson can be interpreted to disclose the specific limitation of a first process receiving a message in a data representation language from a second process. Thus, Johnson cannot be said to anticipate claim 34. The rejection of claim 34 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claim 50.

Section 103(a) Rejection:

The Examiner rejected claim 5 under 35 U.S.C. § 103(a) as being unpatentable over Johnson in view of Gillam ("Java Liaison"). Applicants respectfully traverse the rejection of claim 5 for at least the reasons presented above regarding its independent claim. Applicants further traverse this rejection for at least the following reasons.

Johnson in view of Gillam fails to teach or suggest that the first object comprises **a plurality of instance variables with the same identifier**. The Examiner admits that Johnson fails to teach the limitations of claim 1 and relies on Gillam, citing Gillam's discussion of enumerated types (Gilliam, pages 2-3). Gillam describes an enumerated data type in which an instance of the enumerated type may only have a value from among a set of explicitly defined values. Gillam does not describe an object including a plurality of instance variables with the same identifier. Instead, Gillam describes a method for incorporating C++-style enumerated types in JAVA. Thus, Gilliam describes an enumerated data type in which a plurality of variables of the enumerated type would each have a value from a single pre-defined set of values. Gillam does not teach that a plurality of instance variables with the same identifier.

Gillam, even if combined with Johnson makes no statement at all regarding a plurality of instance variables with the same identifier. Merely describing a new enumerated data type where the value of a variable must be from a defined set of values does not teach or suggest an object including a *plurality of instance variables with the same identifier*.

Johnson in view of Gilliam also fails to teach or suggest that **the entry (in the intermediary table representation of the first object) for each of the plurality of instance variables with the same identifier includes an enumeration value that uniquely identifies the instance variable in the plurality of instance variables with the same identifier**. As noted above, the Examiner admits that Johnson fails to teach the limitations of claim 5, relying on Gillam. However, even if one combines the enumerated data type taught by Gillam with the XMLJavaBean package taught by Johnson, the result would not include an entry in an intermediary table representation of the object (for each of a plurality of instance variables of the same identifier) that includes an enumeration value that uniquely identifies the instance variable in the plurality of instance variables. Instead, combining Gillam's enumerated data type with Johnson's XMLJavaBeans would result in the ability of Johnson's XMLJavaBeans to

load and store objects with instances of Gillam's enumerated data type. But there is not reason at all for a combination of Johnson and Gillam to include in an entry of an intermediary table representation of an object an enumeration value that uniquely identifies the instance variable in a plurality of instance variables with the same identifier.

Furthermore, there would be no need or benefit to including an enumeration value uniquely identifying an instance variable from among each of a plurality of instance variables in an intermediary table representation of an object comprising the plurality of instance variables, as recited in claim 5. The Examiner equates Johnson's Document Object Model (DOM) tree structure as the intermediary table representation of Applicants' claims. Johnson's DOM tree includes various nodes corresponding to different portions of an object being converted to XML. As noted above, the Examiner's combination of Johnson and Gillam would not include a plurality of instance variables with the same identifier. Thus, there would be no reason to include an enumeration value to uniquely identify one of such a plurality of instance variables in Johnson's DOM tree.

Thus, the rejection of claim 5 is not supported by the cited art and removal thereof is respectfully requested.

Regarding all the §102 and §103 rejections above, Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and prompt notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-72000/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: April 23, 2007